Application Development

Component Architecture

Revision Date: September 22, 2000

Table of Contents

Section 1 - Background and Decision Tools

Business Direction

Business Direction, which includes Business Influences, Goals and Objectives, forms the foundation of the BEAM process and the DMV's Enterprise Architecture. This foundation is the first step from which all information technology decisions are made and can be traced. Business objectives are common across the DMV enterprise and represent DMV's stated direction for fulfilling the organization's mission. The primary objective of the BEAM process is to develop a flexible, comprehensive, maintainable framework to manage the rapid evolution of technologies that support the business directions of the DMV. This approach will directly link all technologies implemented to

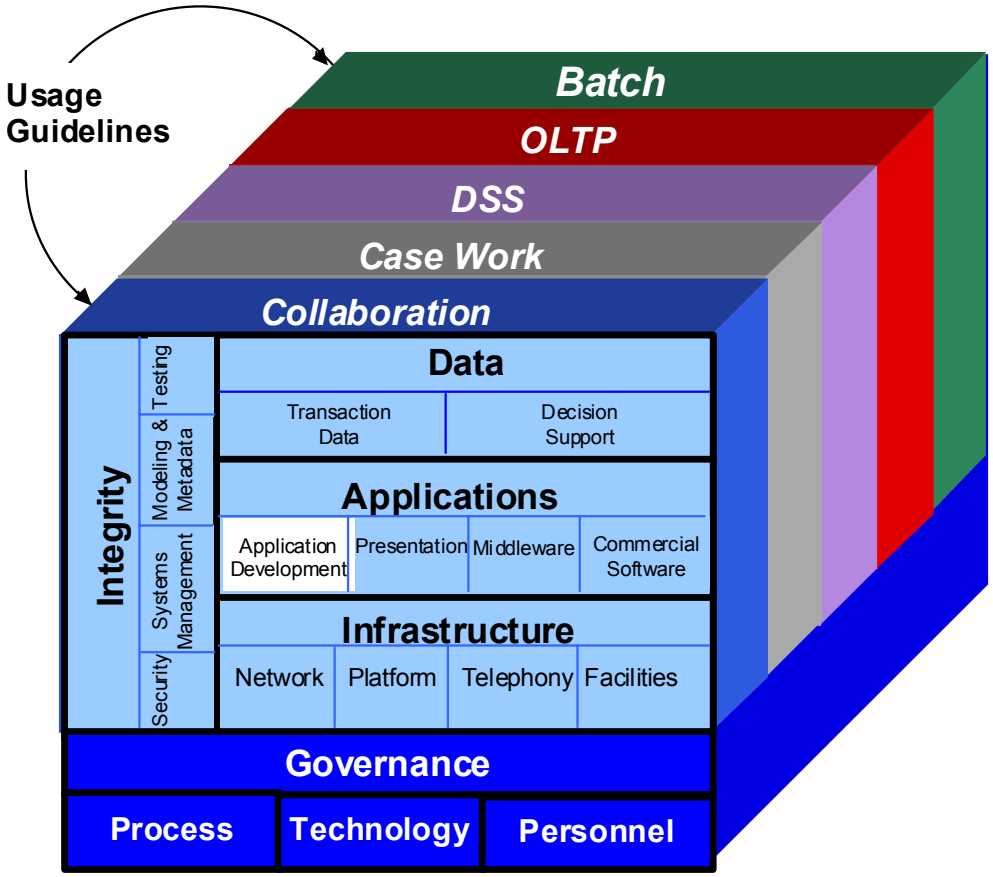
Architecture Requirements

The Architecture Requirements consists of two sections, Information Requirements and Technology Requirements. This is the first step in the BEAM process that begins to focus on specific technologies.

Information Requirements represent the informational needs that are necessary to fulfill DMV's Business Goals and Objectives. Information Requirements bridge the gap between what the Business Goals and Objectives are and what DMV's information systems must deliver to allow management to met these goals and objectives. An individual Information Requirement is typically applicable to more than one Business Objective. Information Requirements are not system or division specific, rather they are related to the information itself. Information delivery refers to the process of delivering information to and from people or groups of people, rather than the input or output of data from databases or applications.

Technology Requirements represent the technologies that satisfy the Information Requirements. In addition, the Technology Requirements feed the DMV Domain Model (Figure 1), which organizes each IT technology into manageable categories called domains, components and Sub-Components. The Technology Requirements detail the specific technologies that satisfy the Information Requirements.

Figure 1: Domains and Components



Conceptual Architecture

The Conceptual Architecture defines the principles and industry-leading best practices that will guide future IT and process decisions. The Conceptual Architecture is derived from DMV's Business Goals, Objectives, Information Requirements and Technology Requirements. Conceptual Architecture Principles (CAPs) are important to the BEAM process because they help ensure that the decisions made later in the Component Architecture development process are consistent.

The following pages contain CAPs that apply to the Application Development Component:

Applicable Conceptual Architecture Principles	
Gov-Proc 6	Follow a total cost of ownership (TCO) methodology.
Gov-Proc 7	Develop service level agreements (SLA) for all IT services.
Gov-Tech 1	Design systems (i.e., hardware, software, operating systems, networks) to be robust enough to handle changing business needs.
Gov-Tech 2	Deploy information systems across an N-Tier*, distributed computing environment.
Gov-Tech 3	Design flexibility into the architecture to accommodate continuing business changes and improvements in technology.
Gov-Tech 4	Design DMV systems for scalability and increased functionality.
Gov-Tech 9	DMV must base all enterprise system development on a set of system independent enterprise design products including an enterprise data model, an enterprise object model, an enterprise business process model and an enterprise business rules repository.
Gov-Tech 10	DMV must not allow individual system development or implementation efforts to dictate enterprise architecture, enterprise data models, enterprise object models, enterprise business process models or enterprise business rules without an independent review from an enterprise level.
Gov-Tech 11	All system development and COTS systems implementations must be fully tested by an independent testing unit prior to deploying the initial system or any major upgrades.
Gov-Pers 13	DMV must use a strong, experienced project manager on every project.
Apps 1	Centrally manage and administer DMV applications.
Apps 2	All business logic will reside in a middle tier, separated from the database access and presentation services.
Apps 3	Document, maintain and manage application component information in a shared enterprise repository.
Apps 4	Represent business logic and transaction services in universally accessible components.
Apps 5	Coordinate and manage business rules from an enterprise perspective.
Apps 6	Business units are responsible for defining and maintaining the integrity of the business rule content within their program area.
Apps 7	Architect systems to be business event driven.
Apps 8	Use packaged solutions where feasible before building custom solutions.
Apps 9	Develop and deploy applications to utilize a common and shared set of server, network and middleware services.
Apps 10	All software development efforts shall include the Project Management Plan.
Apps 11	Utilize a standard system development methodology.

*For a definition of an N-Tier environment, please see the Glossary section of the BEAM intranet site at: http://dmvweb/isd/beam/

Section 2 - BEAM Recommendations

Application Development Component

The Application Development Component provides a guideline for the tools, standards and technologies that will be utilized within the framework of the architecture.

Specific technology Sub-Components in the Application Development component include:

- Requirements Management and Tracking Tools Sub-Component
- Languages Sub-Component
- Code Generators Sub-Component
- Issue Tracking and Resolution Tools Sub-Component
- Version Control Tools Sub-Component
- Integrated Development Environment (IDE) Sub-Component

Application Development Component Principles

The following principles form the core tenets of the Application Development Component

#	Application Development Component Principles
1.	Store business rules in a central enterprise repository, accessible to all authorized users.
2.	Involve users in application development, test and deployment to improve ease of learning, use and support.
3.	Design applications for ease of development, test, deployment, maintenance and support.
4.	Develop application functionality that can be called as services implementing documented component interfaces.
5.	Build reusable components to be shared across the enterprise.
6.	Maintain information about available reusable components.
7.	Ensure that new components do not replicate existing components' functionality.
8.	Utilize effective component management methodologies, including the tools to promote component reuse.
9.	Assign ownership and maintenance responsibilities for all components.
10.	Implement one business rule or function, or a single set of related business rules or functions, per component.
11.	Applications should access existing data prior to prompting users to re-enter data.
12.	Utilize Requirements Management Tools for capturing and tracking application requirements.
13.	Trace requirements throughout the application development lifecycle.
14.	The development environment should support remote, collaborative, multi-organizational and multi-project team development.
15.	Implement enterprise-wide version control processes and procedures.

#	Application Development Component Principles
16.	Conduct performance and volume testing on all applications prior to deployment.

Application Development Toolset Vision*

To provide the speed and flexibility required to deliver solutions to meet the ever-changing demands on the DMV, it is imperative that the applications development tools environment be optimized for, and integrated with, the applications deployment environment. Therefore, to a large degree, the tools for developing applications and managing the application development lifecycle will be driven by the decisions made in the applications, data, and infrastructure areas. That is, the most appropriate applications development toolsets will be those that work best with the standards and products that have been selected for platforms, networks, transaction data, and middleware.

The vision for the applications deployment environment at the DMV is a heterogeneous, multi-tiered, distributed computing environment. Applications will be deployed as discrete, service-based components across (for example) OS390, Solaris and AIX platforms against legacy, and commercial RDBMS data sources (such as DB2 and/or Oracle). These components will be integrated through the use of a middleware solution (Enterprise Application Integration product suite) and target a web browser as the primary client through a shared set of web server services.

To support this, the applications development languages and tools should be selected to coordinate the building and deploying of applications in this complex environment. In addition, the various development and development management tools should integrate their capabilities in order to further simplify these development tasks.

NOTE: The BEAM team recognizes the need to also provide development environment support for batch processing on the mainframe; however, additional research remains to be done before standards and products can be defined for this environment.

There are primarily two language sets targeted for the heterogeneous environment. Java has been adopted by a large number of vendors and ported to most platforms. Most middleware service products and application integration environments support it and IBM has made strong commitments to its future as the preferred language for the mainframe. This is important in the DMV context, since it allows a choice of platforms (OS/390 as well as UNIX) to host mid-tier functionality such as middleware and business logic. This architectural flexibility permits a wide range of performance/cost tradeoffs when implementing applications. For the other alternative, Microsoft has recently unveiled its .NET initiative. Part of that initiative is a new set of development languages / environments. C# (C Sharp) is a modification of the C++ language to add

Java like safety features (garbage collection, etc.) The .NET technologies are extremely new and unproven and are currently primarily restricted to the Windows family platforms. Therefore, Java will be selected as the targeted development language for new development (although maintenance and tweaking of existing systems may still continue in other languages.)

Therefore, the applications development toolset should provide for Java development across the various platforms in the target environment. In addition, given the complexity of designing and building in the distributed computing environment, the applications development tools should be packaged as an integrated suite. Of the leading Java development suites in the market, IBM's VisualAge For Java is the clear market leader and has the richest feature set for building Java applications in a mixed environment like the one here. This suite has the added advantage of a powerful Java code generator and support for Cobol and other languages for legacy maintenance tasks. IBM's VisualAge For Java is bundled with WebSphere Studio Application Developer (WSAD), the core development environment from IBM.

Finally, the tools surrounding the management of application development tasks (requirements management, system modeling, OO design, issue tracking, etc.) need to be streamlined and efficient to give the development lifecycle the speed and flexibility that is required to meet the ever-changing demands on the DMV systems. So it is important these tools provide an integrated set of functions that operate on the various platforms and with the selected development tools.

IBM recommends the Rational suite of products as a best-of-breed solution to work with VisualAge. This suite provides an integrated environment for managing requirements, analysis, design, issue tracking and version control and is a clear market leader. Combined with VisualAge, it provides a unified, end-to-end framework toolset for all aspects of developing and deploying distributed applications.

To summarize, the vision for applications development at DMV is an integrated VisualAge / Rational development environment producing Java based distributed components across a heterogeneous environment.

The subcomponents within the Application Development component, and their related product selections, are:

- Requirements Management and Tracking Tools Sub-Component – Rational Requisite Pro
- Languages Sub-Component – Java (please note that this is the choice for business logic: there are a number of language selections specified for this subcomponent, depending on usage)
- Code Generators Sub-Component – IBM VisualAge For Java and ScriptBuilder.
- Issue Tracking and Resolution Tools Sub-Component – Rational ClearQuest

- Version Control Tools Sub-Component Component – Rational ClearCase Family
- Integrated Development Environment (IDE) Sub-Component – IBM VisualAge For Java

*Note: This Vision section was drafted to serve as the rationale for the Product selections found in the remainder of this document. For this reason, no “Rationale” sections have been provided in the Product selections for each Sub-Component.

Application Development Sub-Components

Requirements Management and Tracking Tools Sub-Component

Definition

Tools to manage requirements, and to support traceability of requirements into design, implementation and documentation artifacts. Requirements management and tracking tools provide for automated support for extracting requirements from a specification, assigning unique identification numbers, tracing requirements through different specification levels, and creating a trace matrix. Requirements can be extracted from a source specification, or manually entered by the user, for storage in a requirements management database. Attributes and keywords can be assigned to each requirement statement to ensure desirable management abilities.

Standards

#	Requirements Management and Tracking Tools Standards
1.	Must exchange data with Microsoft Word for documentation. **Rationale:** • There are no de facto standards for these tools. However, a tool which supports Word format will produce output which is widely accessible without incurring high client-side licensing costs, and can be saved as HTML if required for Web use.
2.	Must support a browser-based client interface **Rationale:** • There are no de facto standards for these tools. A tool, which supports a browser interface, will simplify configuration management, and reduce client-side licensing costs.
3.	Must integrate with DMV standard design tools (e.g., object-oriented modeling tools and data design tools). **Rationale:** The use of DMV standard productivity tools will enable collaborative efforts.

Products

#	Products
1.	Rational RequisitePro

Tech Watch

None

Review Cycle

1 year

Languages Sub-Component

Definition

A language is used to write instructions for the computer. It lets the programmer express data processing in a symbolic manner without regard to machine-specific details.

The statements that are written by the programmer are called code or source code, and they are translated into the computer's machine language by programs called assemblers, compilers and interpreters.

Programming languages fall into two categories: low-level assembly languages and high-level languages. Assembly languages are available for each CPU family, and each assembly instruction is translated into one machine instruction by the assembler program. With high-level languages, a programming statement may be translated into one or several machine instructions by the compiler or may be interpreted at run-time.

Standards

#	Languages Standards
1.	**Standard:** Structured Query Language (SQL) **Usage:** • Language for database queries **Rationale:** • SQL is an industry standard, as well as an ANSI standard (X3.135-1992, soon to be superseded by ANSI/IEC 9075), supported by a large number of vendors. Use of a standard query language eliminates dependence on proprietary query languages [CAP 22, CAP 47].
2.	**Standard:** Stored Procedure Language: • **PL/SQL** • **Transact-SQL** - a Microsoft product compatible with SQL Server • **Java** - supported by recent releases of DB/2 **Usage:** • Procedural language for SQL **Rationale:** • Procedural database management system (DBMS) languages should be used wherever possible to standardize and simplify the processing associated with database transactions. They should be used instead of existing 3GLs (e.g., C) routines since, in general, they are more

	maintainable and self-documenting [CAP 22, CAP 47]. ANSI is developing a standard for procedural extensions of SQL; the first draft of this standard was directly based on PL/SQL. *(Note: Product selection will depend on DBMS selection)*
3	**Standard:** Java **Usage:** • High-level language for business logic programming **Rationale:** • Java is a de facto standard language, controlled by Sun Microsystems. Use of client-side Java enables component reuse (through the OO structure of Java and the use of Java Beans), web-enabled processing (User Interface [UI] components can be served to a browser rather than installed on each user workstation, simplifying configuration management) and standard user-interface components (through the use of Swing, a GUI interface). Use of server-side Java supports multi-threading, which allows concurrent user access [CAP 50], better scalability [CAP 16], and component reuse (Enterprise Java Beans). This solution provides a means of separating database functionality from the UI [CAP 14, CAP 45] and can simplify redistribution of processing among servers [CAP 15]. In addition, the related component architecture (EJB) exposes method interfaces transparently (though to varying extents), which supports reuse and documentation.
4	**Standard:** Application Scripting language: • Perl • Tcl *(Note: Must be platform-independent, should be object-oriented, and compatible with platforms and OS's specified in Infrastructure domain.* *Note further that, for system-level scripting, the tools to be used depend on what is provided with the operating system.)* **Usage:** • Development, test, integration and prototyping. **Rationale:** • Object-oriented scripting languages enable reuse of functionality within scripts. The use of scripting languages simplifies development, test, and the integration of applications, particularly of commercial off-the-shelf (COTS) packages. This improves flexibility when configuring packaged solutions [CAP 51]. Many scripting languages support platform abstraction (e.g., through Posix calls—ref. ISO/IEC 9945-1), which simplifies the task of moving from one platform/OS to another [CAP 13, CAP 15,

#	Languages Standards
	CAP 16] and reduces the present tight coupling of hardware and OS features to applications [CAP 21, CAP 22].
5	**Standard:** There is no standard specified for system-level coding. **Rationale:** • The tools to be used depend on what is supported by the target operating system. **Usage:** • System-level coding only.
6	**Standard:** Hypertext Markup Language (HTML) **Usage:** • Used for editing of Web pages **Rationale:** • This is the standard language used for Web development. It has been standardized by the Word Wide Web Consortium (W3C).
7	**Standard:** JavaScript **Usage:** • Language used for client-side scripting in Web interfaces **Rationale:** • This is the standard language used for Web development. It has been standardized by the W3C.
8	**Standard:** Dynamic HTML **Usage:** • Used for generating dynamic (changing) HTML content **Rationale:** • This is the standard language used for Web development. It has been standardized by the W3C.
9.	**Standard:** eXtensible Markup Language (XML)

#	Languages Standards
	Usage: • XML allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. XML is a format for the interchange of metadata. **Rationale:** • XML is a subset of Standard Generalized Markup Language (SGML) which is designed especially for Web documents. XML may eventually replace HTML as the standard Web formatting specification, but that depends on whether or not it is supported by future Web browsers. Microsoft Internet Explorer version 5 handles XML, but renders it as Cascading Style Sheets (CSS), and Netscape is still experimenting with XML support. In addition, XML tags and schemas are not yet defined for the types of transactions that are applicable at DMV.
10	**Standard:** Enterprise JavaBeans (EJB) **Usage:** • Component/container model **Rationale:** • EJB is consistent with DMV's server platform and business logic language standards.

Tech Watch

None

Review Cycle

1 year

Code Generators Sub-Component

Definition

Code Generators generate application programs from descriptions of the problem rather than by traditional programming. It is at a higher level and easier to use than a high-level programming language. One statement or descriptive line may generate a huge routine or an entire program. However, application generators always have limits as to what they can be used for. Generators used for complex program development allow if-then-else programming to be expressed along with the simpler descriptive entries.

Standards

#	Code Generators Standards
1.	Must generate DMV-supported programming language(s). **Rationale:** • This standard ensures that code generated by code generators does not require special skills or tools to support.

Products

At the time of this review, the following product(s) were identified as the leaders in this sub-component area. A single product has not been select as the DMV standard. When a product is needed, please contact a BEAM representative to further assist in the research and selection process for the department.

#	Products
1.	IBM VisualAge For Java
2.	ScriptBuilder, by NetObjects (for script generation)

Tech Watch

None

Review Cycle

1 year

Issue Tracking and Resolution Tools Sub-Component

Definition

Tools that use a case-tracking mechanism to document and track open issues. Tools for managing the on-going issues and problems associated with test or production activities. In application development, these tools typically are associated with application debugging and testing. Note that these tools do not replace help-desk tools (for example, Remedy), but supplement them.

Standards

#	Issue Tracking and Resolution Tools Standards
1.	Must have a Web-based user interface **Rationale:** • This enables participation in the issue resolution and tracking process without the need to install client software.
2.	Must integrate with the DMV standard version control system. **Rationale:** • Changes to DMV systems should only be made in response to approved bug reports or enhancement requests. This requires integration of check-in/check-out with these reports.
3.	Must allow use of the DMV standard DBMS as a repository. **Rationale:** • This approach enables use of DMV's enterprise Decision Support Software (DSS) for Quantitative Process Management (QPM). QPM uses process metrics as an objective basis for system engineering decisions (e.g., average fix time for bugs, number of new problems found per month, number of requirement changes requested per month). This helps focus the development and maintenance efforts to the problems which are of greatest importance to the DMV's business.

Products

#	Products
1.	Rational ClearQuest

Tech Watch

None

Review Cycle

1 year

Version and Configuration Management Tools Sub-Component

Definition

These are developer oriented tools for the management of source code, bitmaps, documents and related files in a large software project. Version-control tools provide a database that is used to keep track of the revisions made to a program by all the programmers and developers involved in it.

Standards

#	Version and Configuration Management Tools Standards
1.	There are currently no standards identified in this area.

Products

#	Products
1.	Rational ClearCase Family

Tech Watch

None

Review Cycle

1 year

Integrated Development Environments (IDEs) Sub-Component

Definition

IDEs are targeted toward the development of enterprise-wide applications. IDEs support the entire application development lifecycle including requirement analysis, design development, testing and deployment. Applications developed using these tools are very scalable and may be targeted to a large number of users on different operating platforms. They can support a three-tiered architecture with varying level of efforts. With GUI-based application development environments, there is a repository that holds the business logic. Specific attributes of IDEs include:

- Support of database processing
- Provide homogenous development environments for both client and server – all tiers of the application are written with the same language
- Support multiple platforms
- Targeted towards developing industrial strength applications which can be deployed across heterogeneous platforms accessing data from different data sources

Standards

#	IDE Standards
1.	The IDE should support DMV-standard languages. **Rationale:** • There are no international or de facto standards for IDEs. *(NOTE: It is likely that DMV will have different IDEs for database and for middle-tier and UI development.)*
2.	An IDE must support interface to the standard Version Control tools. **Rationale:** • There are no international or de facto standards for IDEs. However, integration of IDEs with version control tools is required to support enterprise-wide, standardized configuration management [CAP 46].
3.	Must support the J2EE Platform specification

#	IDE Standards
	Rationale: • The J2EE platform specification is the accepted industry standard that defines a means for ensuring compatibility between middleware services and components. The designation of J2EE as the DMV middleware standard ensures DMV of being able to take advantage of scalable, component-based, object-oriented architectural standards agreed to by most of the information technology industry

Products

#	Products
1.	IBM VisualAge For Java

Tech Watch

None

Review Cycle

1 year

Revision History

August 30, 2000 – Products version approved at Consensus Meeting.

October 9, 2002 – Reviewed and approved for BEAM Intranet.